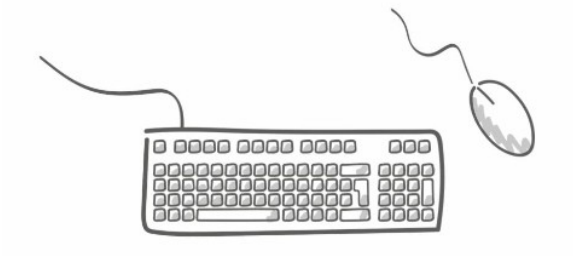
Actions Class

- Nowadays, web applications have functionalities that work only through mouse or keyboard actions.
  Draggable elements, Sliders, Hover Over Dropdowns, Context Click
- To handle those webelements in Selenium, we use Actions class

- The Actions class allows us to **build a chain of actions** that we would like to perform.
  - For example, Press Shift key and type something and then release, or if we want to check multiple checkboxes, we can press CTRL and then click the required checkboxes.



Actions Class

1. First we create an object of Action class
   ```
   Actions act = new Actions(driver);
   ```

2. Then we call methods of action class using this object.

3. **Mouse Actions:**
   click(), clickAndHold(), contextClick(), doubleClick(),
   dragAndDrop(), moveToElement(), release(), build(), perform()

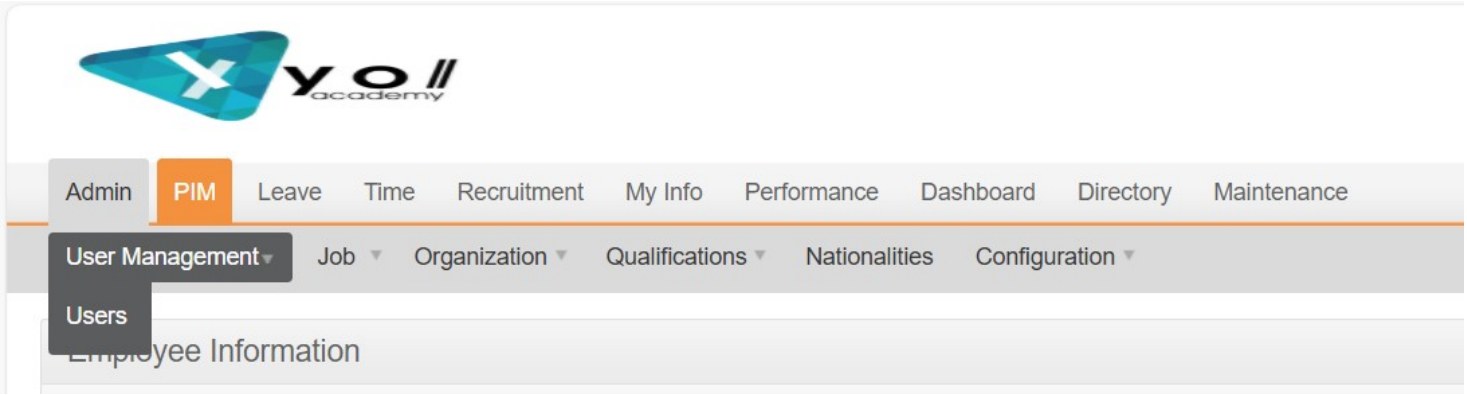
4. **Keyboard Actions**:
   sendKeys(), keyUp(), keyDown()

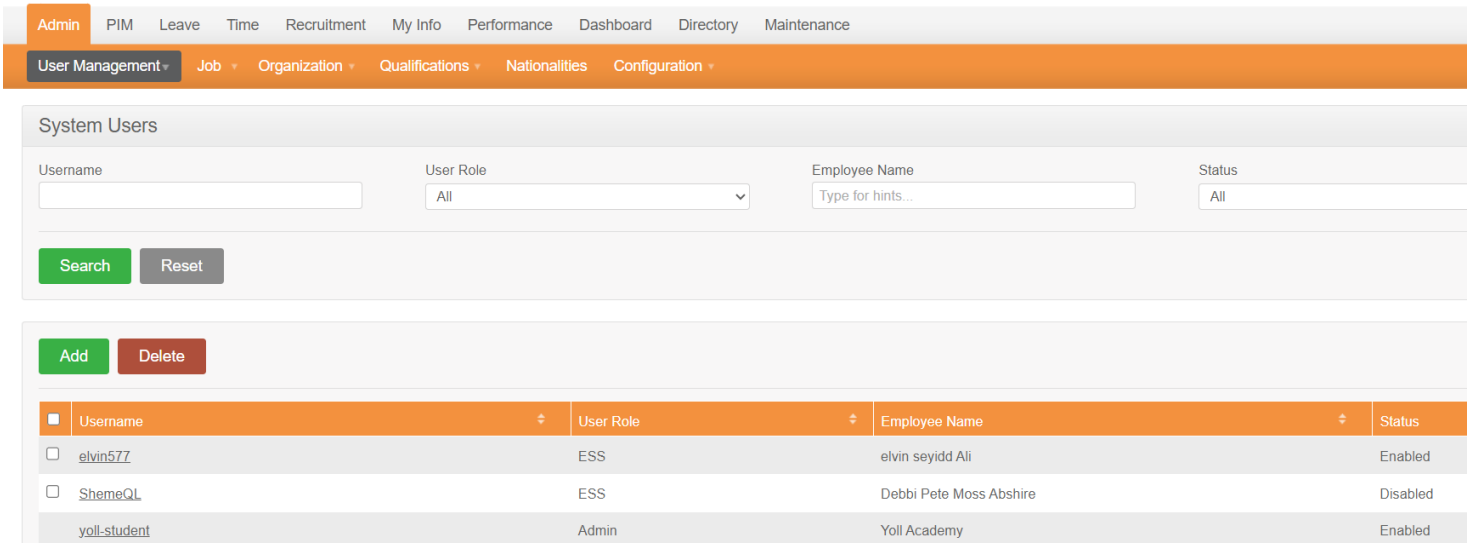
Hover Over

- In order to hover over an element, we use: moveToElement() method.
  - ```
    Action act = new Action(driver);
    act.moveToElement(elem).perform();
    ```
  - Notice we used, **perform()** method at the end, which will execute the action of moveToElement.

Task

```
#  Go to Orange HRM website
#  Login
#  Hover over Admin tab
#  Then hover over to User management
#  Then hover over to users
#  Click on users
```


```




```
Feature: User Management Feature
# Hover over Admin tab
# Then hover over to User management
# Then hover over to users
# Click on users
@run
  Scenario: Tab navigation with actions class
    Given user navigates to hrm login page
    When user logs in with username "yol-student" and password "Bootcamp5#"
    And hovers over the "Admin" tab
    And hovers over the "User Management" sub tab
    And hovers over the "Users" sub tab
    And clicks on Users
    Then user can see page header "System Users"
```

```
package pages;

import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

import java.util.List;

public class HomePage {

    public HomePage(WebDriver driver) {
        PageFactory.initElements(driver, this);
    }
    @FindBy(id = "welcome")
    public WebElement welcomeMessage;
    //@FindBy(xpath = "//*[@id='welcome']")
   // @FindBy(css = "#welcome")
```

```
    @FindBy(xpath = "//div[@class='menu']//b")
    public List<WebElement> tabs;
    @FindBy(id = "menu_pim_viewPimModule")
    public WebElement pimTab;
    @FindBy(id = "menu_admin_UserManagement")
    public WebElement userManagementSubTab;
    @FindBy(id = "menu_admin_viewSystemUsers")
    public WebElement usersSubTab;
    @FindBy(id = "menu_admin_Job")
    public WebElement jobSubTab;
    @FindBy(css = ".header>1")
    public WebElement systemUsersPageHeader;


    public void clickOnTabByName(String tabName) {
        for (WebElement tab : tabs) {
            if (tabName.equalsIgnoreCase(tab.getText())) {
                tab.click();
                break;
            }
        }
    }


    public WebElement getTabWebElementByName(String tabName) {
        for (WebElement tab: tabs) {
            if(tabName.equalsIgnoreCase(tab.getText())){
                return tab;
            }}
            throw new NoSuchElementException("No such tab name as" +tabName + "'");

        } }


package step_definitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import pages.HomePage;
import utilities.Driver;
import utilities.DriverFactory;
import utilities.WaitHelper;

import java.time.Duration;
import java.util.List;

public class HomePageSteps {
    private final WebDriver driver = DriverFactory.getDriver("firefox");
   private final HomePage homePage = new HomePage(driver); // Pass WebDriver to HomePage
constructor
   private final WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
```

```
    // 1. Create an Actions class
   private Actions actions = new Actions (Driver.getInstance().getDriver());

    @Then("user is redirected to home page")
    public void user_is_redirected_to_home_page() {
        wait.until(ExpectedConditions.visibilityOf(homePage.welcomeMessage)); // Ensure
element is visible
        String actualWelcomeMessage = homePage.welcomeMessage.getText();
        String expectedWelcomeMessage = "Welcome Yoll";
        Assert.assertEquals("Welcome message verification failed", expectedWelcomeMessage,
actualWelcomeMessage);
    }

    @Then("user can see following tabs:")
    public void user_can_see_following_tabs(List<String> eTabs) {
        wait.until(ExpectedConditions.visibilityOfAllElements(homePage.tabs)); // Ensure
all tabs are visible
        List<String> expectedTabs = eTabs;
        List<WebElement> actualTabs = homePage.tabs;

        // Check tab counts
        Assert.assertEquals("Tab count mismatch!", expectedTabs.size(), actualTabs.size());

        // Validate tab texts
        for (int i = 0; i < actualTabs.size(); i++) {
            String expectedTab = expectedTabs.get(i);
            String actualTab = actualTabs.get(i).getText().trim();
            Assert.assertEquals("Tab mismatch at index " + i, expectedTab, actualTab);
        }
    }


//    @Given("user navigates to hrm login page")
//    public void user_navigates_to_hrm_login_page() {
//        // Write code here that turns the phrase above into concrete actions
//        throw new io.cucumber.java.PendingException();
//    }
//    @When("user logs in with username {string} and password {string}")
//    public void user_logs_in_with_username_and_password(String string, String string2) {
//        // Write code here that turns the phrase above into concrete actions
//        throw new io.cucumber.java.PendingException();
//    }
    @When("hovers over the {string} tab")
    public void hovers_over_the_tab(String tabName) throws InterruptedException {
        WebElement adminTab=homePage.getTabWebElementByName(tabName);

// hover over the tab
// 1. Create an Actions class
        Actions actions = new Actions (Driver.getInstance().getDriver());
// hover over is achieved by moveToElement (WebElement w) method
        actions.moveToElement(adminTab).perform();
        Thread.sleep( 5000);
    }
    @When("hovers over the {string} sub tab")
    public void hovers_over_the_sub_tab(String subTabName) {
        // First Method
//        if (subTabName.equals("User Management")){
//            actions.moveToElement(homePage.userManagementSubTab).perform();
```

```
//        } else if (subTabName.equals("Job"))
//        {
//            actions.moveToElement(homePage.jobSubTab).perform();
//        }else if (subTabName.equals("Users"))
//        {
//            actions.moveToElement(homePage.usersSubTab).perform();
//        }
        // Second Method

        switch (subTabName) {
            case "User Management" :
                actions.moveToElement(homePage.userManagementSubTab).perform();
                WaitHelper.shortWait();
                break;

            case "Job" :
                actions.moveToElement(homePage.jobSubTab).perform();
                WaitHelper.shortWait();
                break;


            case "Users" :
                actions.moveToElement(homePage.usersSubTab).perform();
                WaitHelper.shortWait();
                break;
        }
    }
    @When("clicks on Users")
    public void clicks_on_users() {
    actions.click().perform();
    WaitHelper.shortWait();
    }
    @Then("user can see page header {string}")
    public void user_can_see_page_header(String expectedPageHeader) {

        String actualPageHeader= homePage.systemUsersPageHeader.getText();
        Assert.assertEquals(  "Page header verification failed!", expectedPageHeader,
actualPageHeader);
    }
}


package runner;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/resources/features",
        glue = "step_definitions",
        tags = "@run",
        dryRun = true,
        monochrome = true,
        plugin = {"pretty", "html:target/reports/cucumber-reports.html"}
)
public class TestRunner {
```

```
    // No code is needed here; Cucumber will handle the execution
}
```

```
package utilities;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.*;
import java.time.Duration;

public class WaitHelper {

    private static final Logger oLog = LogManager.getLogger(WaitHelper.class);


    // Short Wait:

    public static void shortWait() {
        System.out.println("Short wait has been used");
        try {
            Thread.sleep(5008);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    // Hard wait using Thread.sleep()
    public static void wait(int secs) {
        try {
            Thread.sleep(1000L * secs);
        } catch (InterruptedException e) {
            oLog.error("Thread interrupted during sleep", e);
            Thread.currentThread().interrupt(); // Restore interrupted state
        }
    }

    // Wait until an element is clickable
    public static void waitUntilClickable(WebElement element, int timeToWaitInSec) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeToWaitInSec));
            wait.until(ExpectedConditions.elementToBeClickable(element));
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for element to be clickable: " + element, e);
        }
    }

    // Wait until an element is visible
    public static WebElement waitForVisibility(WebElement element, int timeToWaitInSec) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeToWaitInSec));
            return wait.until(ExpectedConditions.visibilityOf(element));
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for element visibility: " + element, e);
            throw e;
        }
    }
```

```
    // Wait for an element to be clickable and return it
    public static WebElement waitForClickability(WebElement element, int timeout) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeout));
            return wait.until(ExpectedConditions.elementToBeClickable(element));
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for element to be clickable: " + element, e);
            throw e;
        }
    }

    // Wait for the page to load completely
    public static void waitForPageToLoad(long timeOutInSeconds) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeOutInSeconds));
            wait.until(driver -> ((JavascriptExecutor) driver).executeScript("return
document.readyState").equals("complete"));
            oLog.info("Page loaded successfully.");
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for page to load after " + timeOutInSeconds + "
seconds", e);
        }
    }

    // Fluent wait for a specific element
    public static WebElement fluentWait(final WebElement webElement, int timeInSec) {
        FluentWait<WebDriver> wait = new FluentWait<>(Driver.getInstance().getDriver())
                .withTimeout(Duration.ofSeconds(timeInSec))
                .pollingEvery(Duration.ofMillis(500))
                .ignoring(NoSuchElementException.class);
        return wait.until(driver -> webElement);
    }

    // Hard wait in milliseconds
    public void hardWait(int timeOutInMillis) throws InterruptedException {
        oLog.info("Hard wait for " + timeOutInMillis + " milliseconds.");
        Thread.sleep(timeOutInMillis);
    }

    // Handle stale element exceptions
    public WebElement handleStaleElement(By locator, int retryCount, int delayInSeconds)
throws InterruptedException {
        oLog.info("Attempting to recover from stale element exception for locator: " +
locator);
        WebElement element = null;

        while (retryCount > 0) {
            try {
                element = Driver.getInstance().getDriver().findElement(locator);
                return element;
            } catch (StaleElementReferenceException e) {
                oLog.warn("StaleElementReferenceException caught. Retrying... Attempts
left: " + (retryCount - 1));
                hardWait(delayInSeconds * 1000);
                retryCount--;
```

```
            }
        }
        oLog.error("Element cannot be recovered after multiple attempts for locator: " +
locator);
        throw new StaleElementReferenceException("Element cannot be recovered");
    }
}
```

```
package utilities;

import org.openqa.selenium.WebDriver;

public class Driver {

    private Driver(){}

    private static Driver driverInstance=new Driver();

    public static Driver getInstance(){
        return driverInstance;
    }

    ThreadLocal<WebDriver> threadLocalDriver=new ThreadLocal<WebDriver>();

    public WebDriver getDriver(){
        WebDriver driver = threadLocalDriver.get();
        return driver;
    }

    public  void setDriver(WebDriver driverParameter){
        // set the parameter driver into threadLocal
        threadLocalDriver.set(driverParameter);
    }

    public void removeDriver(){
        WebDriver driver = threadLocalDriver.get();
        driver.quit();
        threadLocalDriver.remove();
    }
}
```

```
package utilities;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.safari.SafariDriver;

public class DriverFactory {
    public static WebDriver getDriver(String browserType){
        return switch(browserType){
            case "fireFox" ->  new FirefoxDriver();
            case "safari" -> new SafariDriver();
            default ->  new ChromeDriver();
        };
    }
```

```
}
```

```
package utilities;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.*;
import java.time.Duration;

public class WaitHelper {

    private static final Logger oLog = LogManager.getLogger(WaitHelper.class);


    // Short Wait:

    public static void shortWait() {
        System.out.println("Short wait has been used");
        try {
            Thread.sleep(5008);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    // Hard wait using Thread.sleep()
    public static void wait(int secs) {
        try {
            Thread.sleep(1000L * secs);
        } catch (InterruptedException e) {
            oLog.error("Thread interrupted during sleep", e);
            Thread.currentThread().interrupt(); // Restore interrupted state
        }
    }

    // Wait until an element is clickable
    public static void waitUntilClickable(WebElement element, int timeToWaitInSec) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeToWaitInSec));
            wait.until(ExpectedConditions.elementToBeClickable(element));
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for element to be clickable: " + element, e);
        }
    }

    // Wait until an element is visible
    public static WebElement waitForVisibility(WebElement element, int timeToWaitInSec) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeToWaitInSec));
            return wait.until(ExpectedConditions.visibilityOf(element));
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for element visibility: " + element, e);
            throw e;
        }
    }
```

```
    // Wait for an element to be clickable and return it
    public static WebElement waitForClickability(WebElement element, int timeout) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeout));
            return wait.until(ExpectedConditions.elementToBeClickable(element));
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for element to be clickable: " + element, e);
            throw e;
        }
    }

    // Wait for the page to load completely
    public static void waitForPageToLoad(long timeOutInSeconds) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeOutInSeconds));
            wait.until(driver -> ((JavascriptExecutor) driver).executeScript("return
document.readyState").equals("complete"));
            oLog.info("Page loaded successfully.");
        } catch (TimeoutException e) {
            oLog.error("Timeout waiting for page to load after " + timeOutInSeconds + "
seconds", e);
        }
    }

    // Fluent wait for a specific element
    public static WebElement fluentWait(final WebElement webElement, int timeInSec) {
        FluentWait<WebDriver> wait = new FluentWait<>(Driver.getInstance().getDriver())
                .withTimeout(Duration.ofSeconds(timeInSec))
                .pollingEvery(Duration.ofMillis(500))
                .ignoring(NoSuchElementException.class);
        return wait.until(driver -> webElement);
    }

    // Hard wait in milliseconds
    public void hardWait(int timeOutInMillis) throws InterruptedException {
        oLog.info("Hard wait for " + timeOutInMillis + " milliseconds.");
        Thread.sleep(timeOutInMillis);
    }

    // Handle stale element exceptions
    public WebElement handleStaleElement(By locator, int retryCount, int delayInSeconds)
throws InterruptedException {
        oLog.info("Attempting to recover from stale element exception for locator: " +
locator);
        WebElement element = null;

        while (retryCount > 0) {
            try {
                element = Driver.getInstance().getDriver().findElement(locator);
                return element;
            } catch (StaleElementReferenceException e) {
                oLog.warn("StaleElementReferenceException caught. Retrying... Attempts
left: " + (retryCount - 1));
                hardWait(delayInSeconds * 1000);
                retryCount--;
```

```
            }
        }
        oLog.error("Element cannot be recovered after multiple attempts for locator: " +
locator);
        throw new StaleElementReferenceException("Element cannot be recovered");
    }
}
```

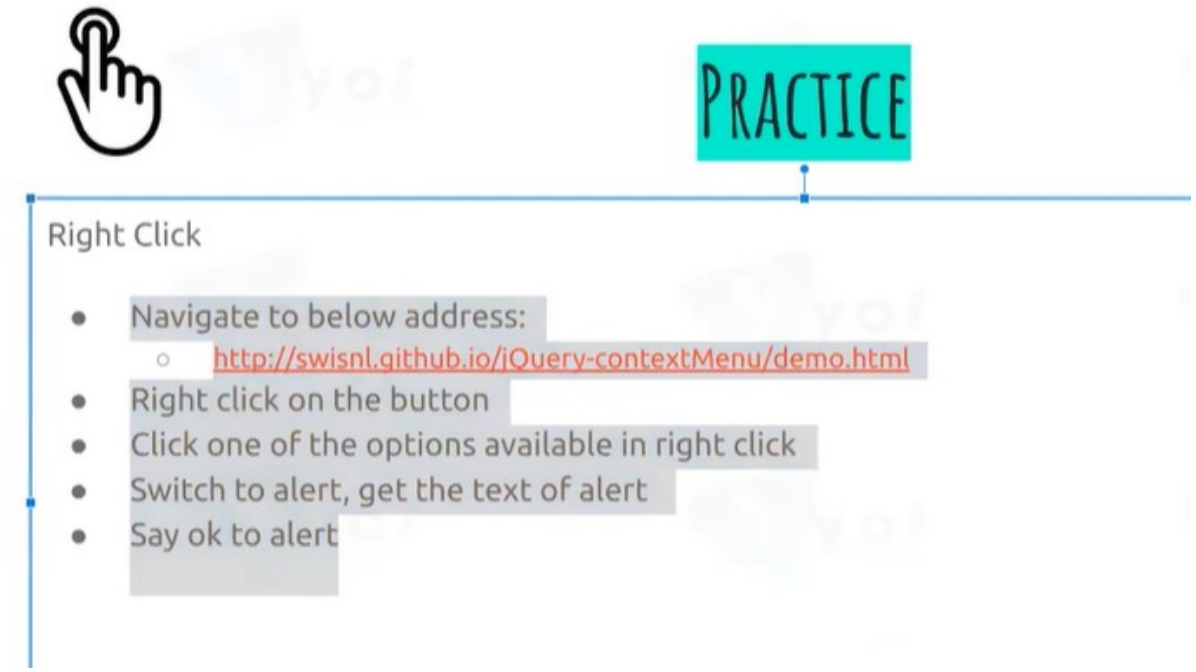
Right Click, Drag and Drop

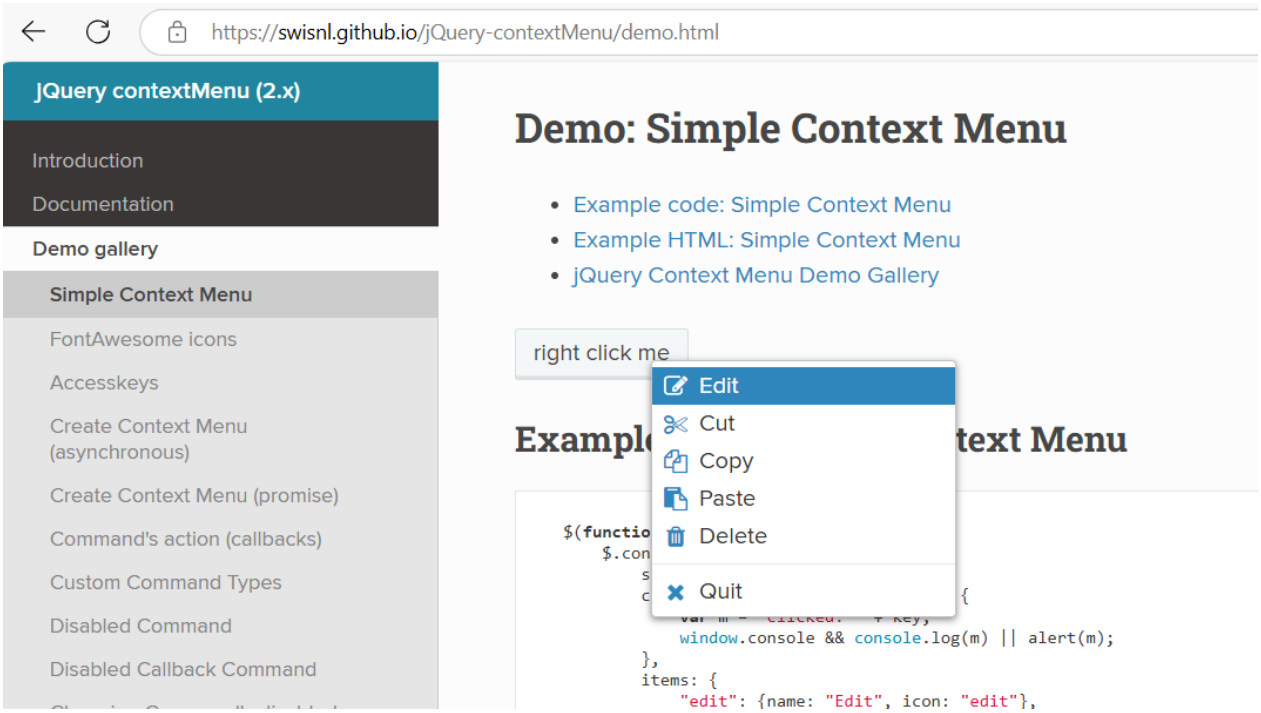
- In order to right click an element or an webpage, we use **contextClick()** method
  - ```
    Actions act = new Actions(driver);
    act.contextClick(element).perform();
    ```

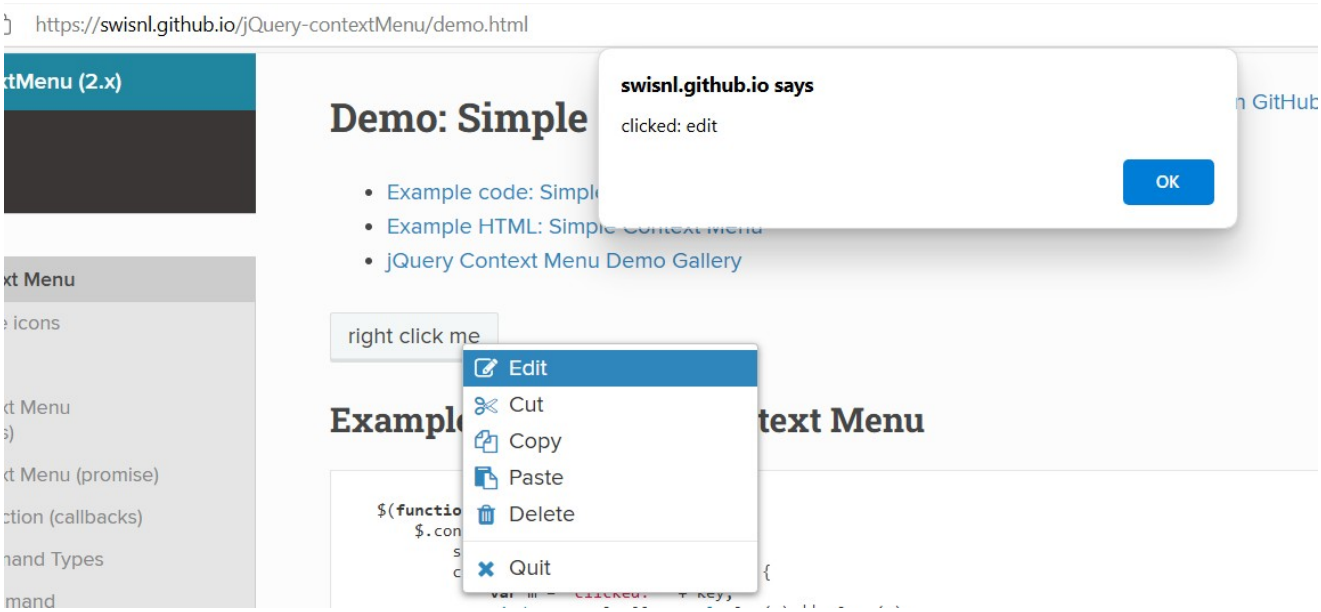
- In order to drag and drop element we use **dragAndDrop()** method
  - ```
    WebElement source = driver.findElement(By.xpath("//img"));
    ```
  - ```
    WebElement target = driver.findElement(By.id("div2"));
    ```
  - ```
    act.dragAndDrop(source, target).perform();
    ```

BUILD()

- If we want to build series of actions methods, we use build() method
  - ```
    act
        .moveToElement(txtUsername)
        .click()
        .keyDown(txtUsername, Keys.SHIFT)
        .sendKeys(txtUsername, "hello")
        .keyUp(txtUsername, Keys.SHIFT)
        .build().perform() ;
    ```
    ```


PRACTICE
Right Click
Navigate to below address:
http://swisnl.github.io/jQuery-contextMenu/demo.html
Right click on the button
Click one of the options available in right click
Switch to alert, get the text of alert
Say ok to alert


https://swisnl.github.io/jQuery-contextMenu/demo.html
JQuery contextMenu (2.x)
Introduction
Documentation
Demo gallery
Simple Context Menu
FontAwesome icons
Accesskeys
Create Context Menu (asynchronous)
Create Context Menu (promise)
Command's action (callbacks)
Custom Command Types
Disabled Command
Disabled Callback Command
Demo: Simple Context Menu
Example code: Simple Context Menu
Example HTML: Simple Context Menu
jQuery Context Menu Demo Gallery
right click me
Edit
Cut
Copy
Paste
Delete
Quit
window.console && console.log(m) || alert(m);
},
items: {
"edit": {name: "Edit", icon: "edit"},



First Method

```
Feature: Context Click on JQuery
  @Query
  Scenario: Context click show alert with a message
    Given user navigates to jquery application
    When user right clicks on the right click me button
    And selects "Edit" option from the context menu
    Then user can see an alert with content as "edit"
    And can accept the alert
```

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import utilities.Driver;

public class JQueryPage {

    public JQueryPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);
    }

    @FindBy(css = "span.context-menu-one")
    public WebElement rightClickMeButton;

    public WebElement getContextMenuWebElementByName(String optionName) {
        String xpath = "//ul[contains(@class,'context-menu-list')]//span[text()='" +
optionName + "']";
        return Driver.getInstance().getDriver().findElement(By.xpath(xpath));
```

```
    }
}
```

```
package step_definitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

import org.junit.Assert;
import org.openqa.selenium.Alert;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import pages.JQueryPage;
import utilities.Driver;
import utilities.WaitHelper;

public class JQueryPageSteps {

    private final Actions actions;
    private final JQueryPage jQueryPage;

    public JQueryPageSteps() {
        this.actions = new Actions(Driver.getInstance().getDriver());
        this.jQueryPage = new JQueryPage();
    }

    @Given("user navigates to jquery application")
    public void user_navigates_to_jquery_application() {
        String url = "https://swisnl.github.io/jQuery-contextMenu/demo.html";
        Driver.getInstance().getDriver().get(url);
    }

    @When("user right clicks on the right click me button")
    public void user_right_clicks_on_the_right_click_me_button() {
        WaitHelper.shortWait();
        actions.contextClick(jQueryPage.rightClickMeButton).perform();
        WaitHelper.shortWait();
    }

    @When("selects {string} option from the context menu")
    public void selects_option_from_the_context_menu(String optionName) {
        WebElement option = jQueryPage.getContextMenuWebElementByName(optionName);
        WaitHelper.shortWait();
        actions.click(option).perform();
        WaitHelper.shortWait();
    }

    @Then("user can see an alert with content as {string}")
    public void user_can_see_an_alert_with_content_as(String text) {
        Alert alert = Driver.getInstance().getDriver().switchTo().alert();
        String actualAlertText = alert.getText();
        Assert.assertTrue("Alert message verification failed",
actualAlertText.endsWith(text));
    }
```

```
    @Then("can accept the alert")
    public void can_accept_the_alert() {
        Alert alert = Driver.getInstance().getDriver().switchTo().alert();
        alert.accept();
    }
}


package utilities;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class Driver {

    /* private constructor, no objects can be created outside of this class */
    private Driver(){

    }

    /* private static reference variable to the class */
    private static Driver instance=new Driver();

    /* create a getter method with return type of the class => encapsulation */
    public static Driver getInstance(){
        return instance;
    }
    private static ThreadLocal<WebDriver> threadLocalDriver = new ThreadLocal<>();

    public static WebDriver getDriver() {
        // Initialize the driver if it is null
        if (threadLocalDriver.get() == null) {
            System.setProperty("webdriver.chrome.driver",
"src/test/resources/drivers/chromedriver.exe"); // Set the path to your chromedriver
            ChromeOptions options = new ChromeOptions();
            options.addArguments("--start-maximized");
            threadLocalDriver.set(new ChromeDriver(options));
        }
        return threadLocalDriver.get();
    }

    public static void closeDriver() {
        if (threadLocalDriver.get() != null) {
            threadLocalDriver.get().quit();
            threadLocalDriver.remove();
        }
    }



}



package utilities;

import io.cucumber.java.After;
```

```
import io.cucumber.java.Before;

public class Hooks {

    @Before
    public void setUp() {
        // Initialize the WebDriver instance
        Driver.getDriver();
    }

    @After
    public void tearDown() {
        // Close the WebDriver instance
        Driver.closeDriver();
    }
}
```

```
package utilities;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

public class WaitHelper {
    private static final Logger oLog = LogManager.getLogger(WaitHelper.class);

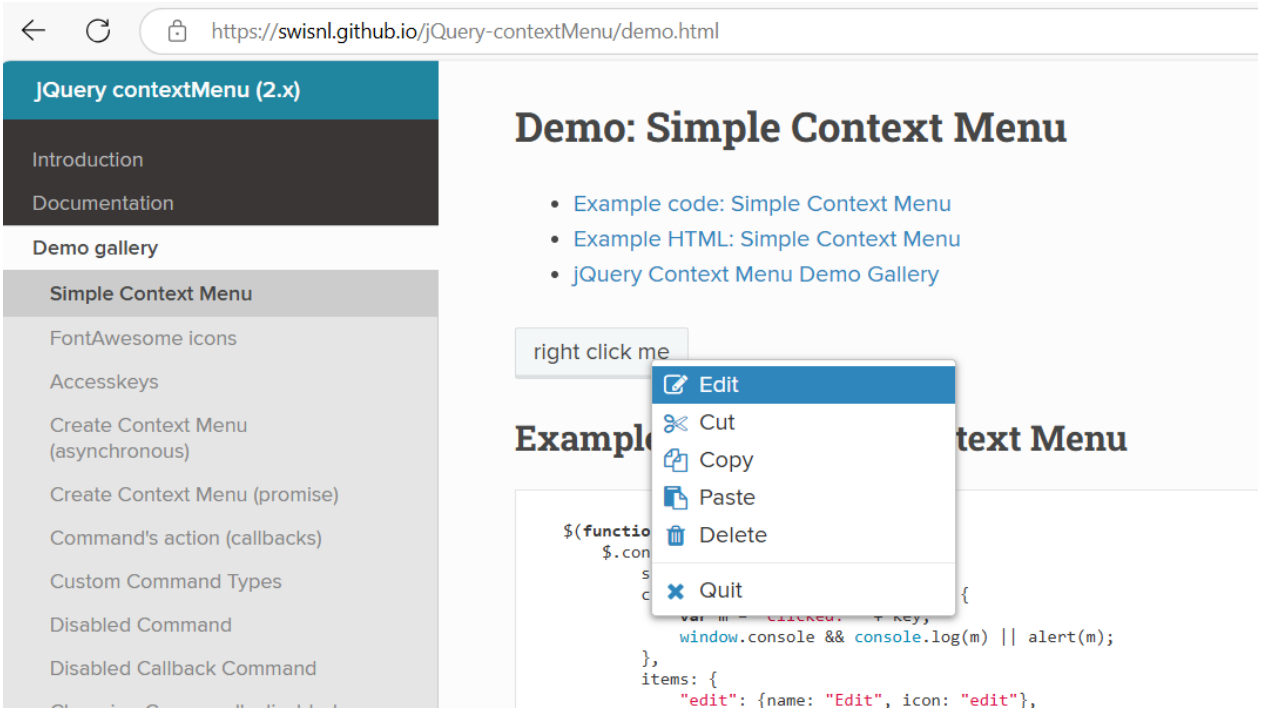
    // Short Wait
    public static void shortWait() {
        System.out.println("Short wait has been used");
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

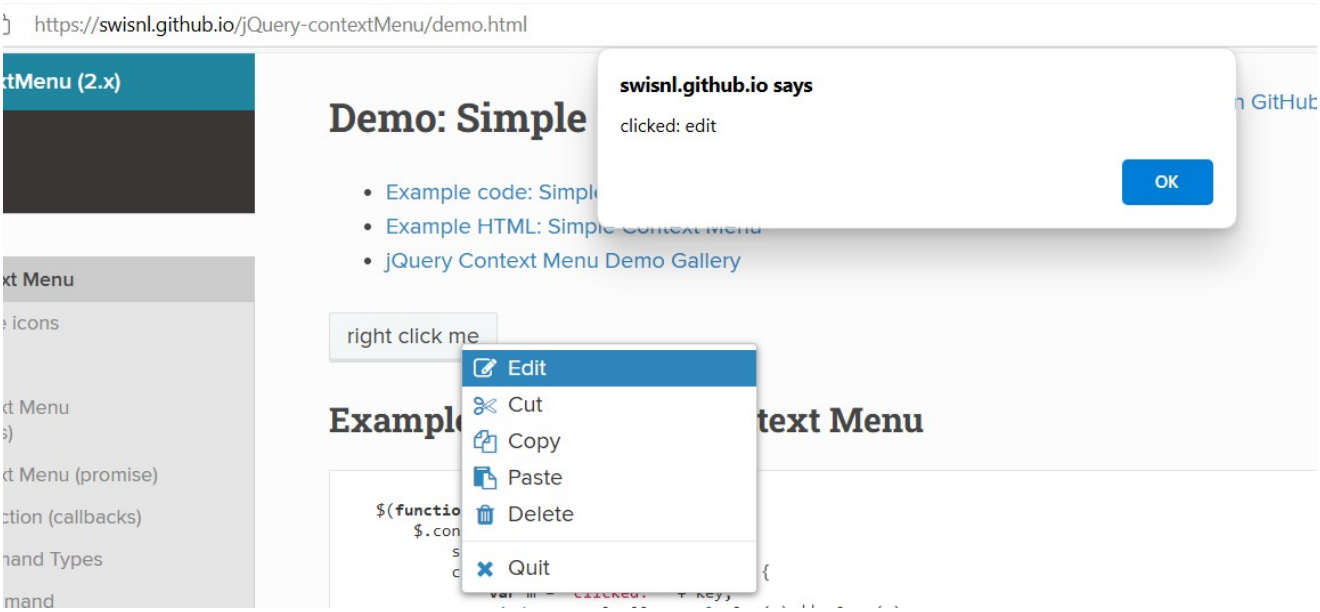
    // Wait until an element is clickable
    public static void waitUntilClickable(WebElement element, int timeToWaitInSec) {
        try {
            WebDriverWait wait = new WebDriverWait(Driver.getInstance().getDriver(),
Duration.ofSeconds(timeToWaitInSec));
            wait.until(ExpectedConditions.elementToBeClickable(element));
        } catch (Exception e) {
            oLog.error("Timeout waiting for element to be clickable: " + element, e);
        }
    }
}
```

Second Method With Outline Scenario:

```
Feature: Context Click on JQuery
  @Query
  Scenario Outline: Context click show alert with a message
    Given user navigates to jquery application
    When user right clicks on the right click me button
    And selects "<Option>" option from the context menu
    Then user can see an alert with content as "<Text>"
    And can accept the alert
    Examples:
    | Option| Text   |
    | Edit  | Edit   |
    | Cut   | Cut    |
    | Copy  | Copy   |
    | Paste | Paste  |
    | Delete| Delete|
    | Quit  | Quit   |
```

Result:





The difference between the first and second method is that second can show:

clicked: `Cut Copy  Paste Delete Quit` **NOT ONLY Edit** as the first method show.

Using Git

Git add – A

Git status

Git commit -m “Actions class addition”

Git pull

Project ies HomePageSteps.java context-menu-actions.feature

Terminal Local

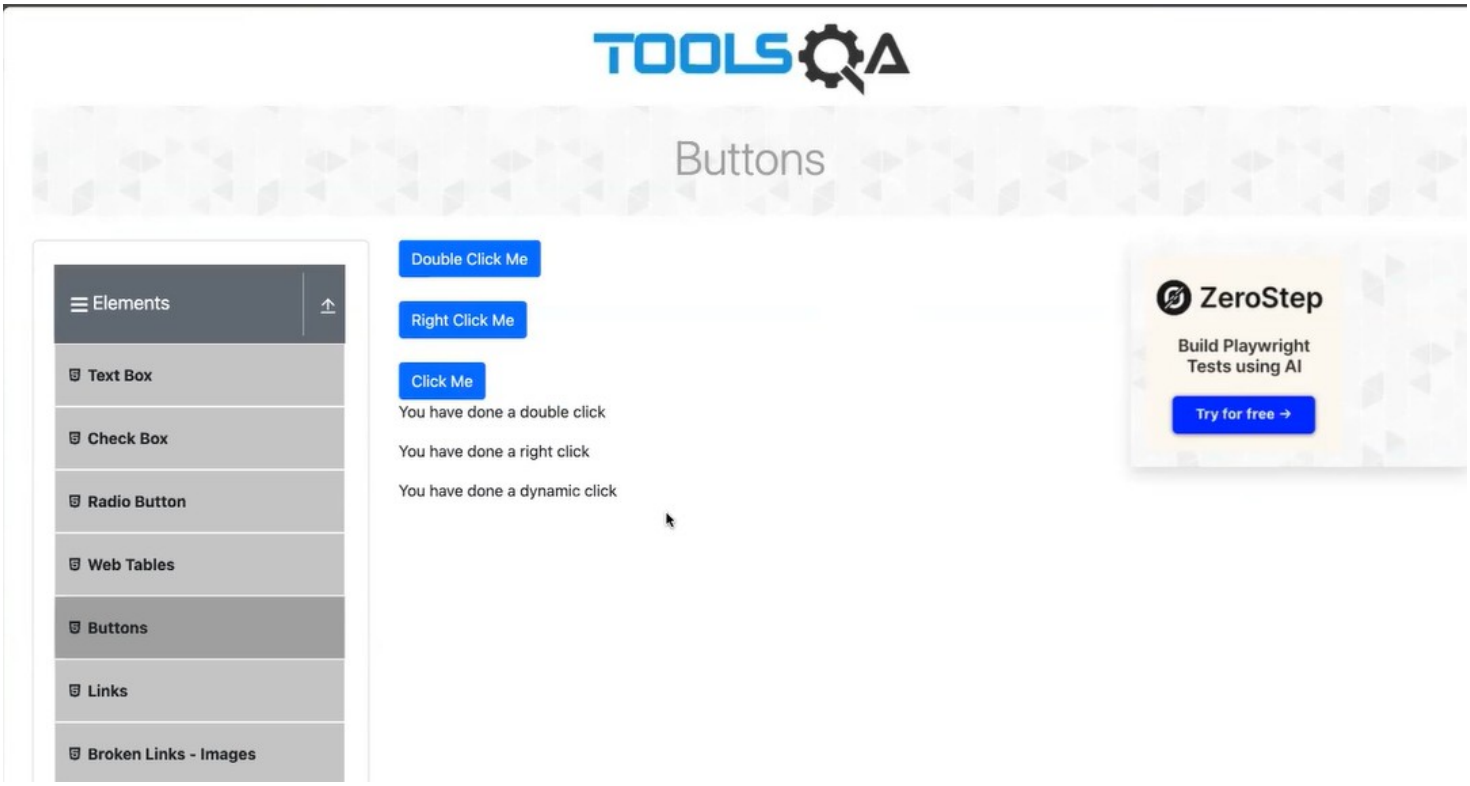
```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        modified:   src/test/java/runners/UITestRunner.java
        new file:   src/test/java/ui_automation/pages/JQueryPage.java
        modified:   src/test/java/ui_automation/pages/OhrmHomePage.java
        modified:   src/test/java/ui_automation/step_definitions/HomePageSteps.java
        modified:   src/test/java/ui_automation/step_definitions/Hooks.java
        new file:   src/test/java/ui_automation/step_definitions/JQueryPageSteps.java
        modified:   src/test/java/ui_automation/utilities/AlertHelper.java
        modified:   src/test/java/ui_automation/utilities/BrowserFactory.java
        modified:   src/test/java/ui_automation/utilities/Helper.java
        modified:   src/test/java/ui_automation/utilities/SelectHelper.java
        modified:   src/test/java/ui_automation/utilities/WaitHelper.java
        modified:   src/test/resources/api-config.properties
        modified:   src/test/resources/ui-config.properties
        new file:   src/test/resources/ui_features/context-menu-actions.feature
        modified:   src/test/resources/ui_features/user-login.feature
        new file:   src/test/resources/ui_features/user-management.feature

→  YFrameworkFinalC13 git:(master) ✗ git commit -m"Actions class addition"
[master 30f6fe4] Actions class addition
 17 files changed, 246 insertions(+), 46 deletions(-)
 create mode 100644 src/test/java/ui_automation/pages/JQueryPage.java
 create mode 100644 src/test/java/ui_automation/step_definitions/JQueryPageSteps.java
 create mode 100644 src/test/resources/ui_features/context-menu-actions.feature
 create mode 100644 src/test/resources/ui_features/user-management.feature
→  YFrameworkFinalC13 git:(master)
```

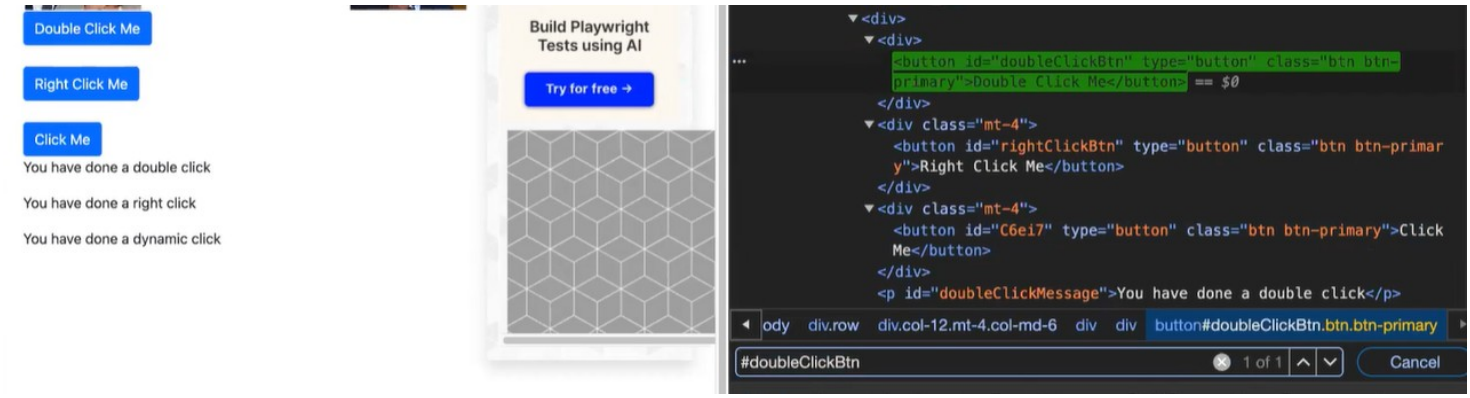
TASK

Right Click

- Navigate to below address:
  - https://demoqa.com/buttons
- Right click on the button
- Verify “You have done a right click” verbiage

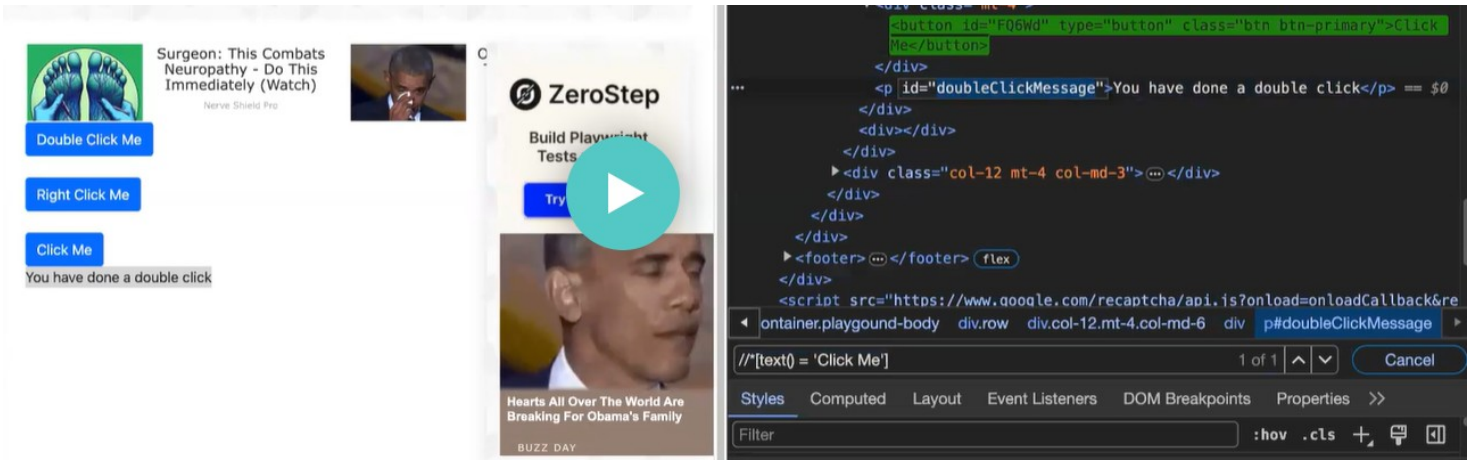


Inspecting the buttons



Inspecting the message:

You have done a double click



To use the RIGHT CLICK: CHOOSE CONTEXTCLICK()

First Method: using only the first button "Double Click Me"

```
Feature: Demo QA Action Buttons
  @actionButtons
  Scenario: Interacting with action buttons
    Given user navigates to demoqa action buttons page
    #When user clicks on "Button" button
    #Then user can see a message "You have done a right click"

      ###First button : Double Click Me
    When user clicks on "Double Click Me" button
    Then user can see a message "You have done a double click"
```

```
package step_definitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.interactions.Actions;
import pages.DemoQaPage;
import utilities.ConfigurationReader;
import utilities.Driver;
import utilities.WaitHelper;

public class DemoQaPageSteps {
    private DemoQaPage demoQaPage = new DemoQaPage();
    private Actions actions = new Actions(Driver.getInstance().getDriver());
    @Given("user navigates to demoqa action buttons page")
    public void user_navigates_to_demoqa_action_buttons_page() {
        Driver.getInstance().getDriver().get(ConfigurationReader.getProperty("ui-
config.properties","demoqa.url"));
        WaitHelper.shortWait();
    }
    @When("user clicks on {string} button")
    public void user_clicks_on_button(String buttonName) {
        actions.click(demoQaPage.doubleClickButton).perform();
        WaitHelper.shortWait();

    }
```

```
    @Then("user can see a message {string}")
    public void user_can_see_a_message(String expectedMessage) {
        String actualMessage = demoQaPage.doubleClickMessage.getText();
        Assert.assertEquals("Message verification failed",expectedMessage,actualMessage);

    }

}
```

Second Method:using Scenario Outline

```
Feature: Demo QA Action Buttons
  @actionButtons
  Scenario Outline: Interacting with action buttons
    Given user navigates to demoqa action buttons page
    When user clicks on "<Button>" button
    Then user can see a message "<Message>"
    Examples:
  | Button          | Message                          |
  |Double Click Me | You have done a double click  |
  | Right Click Me | You have done a right click   |
  | Click Me        | You have done a dynamic click |
```

```
package step_definitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.interactions.Actions;
import pages.DemoQaPage;
import utilities.ConfigurationReader;
import utilities.Driver;
import utilities.WaitHelper;

public class DemoQaPageSteps {
    private DemoQaPage demoQaPage = new DemoQaPage();
    private Actions actions = new Actions(Driver.getInstance().getDriver());
    private String buttonName;
    @Given("user navigates to demoqa action buttons page")
    public void user_navigates_to_demoqa_action_buttons_page() {
        Driver.getInstance().getDriver().get(ConfigurationReader.getProperty("ui-
config.properties","demoqa.url"));
        WaitHelper.shortWait();
    }
    @When("user clicks on {string} button")
    public void user_clicks_on_button(String buttonName) {
//        actions.click(demoQaPage.doubleClickButton).perform();
//        WaitHelper.shortWait();
        this.buttonName=buttonName;
        switch (buttonName) {
            case "Double Click Me":
                actions.doubleClick(demoQaPage.doubleClickButton).perform();
                break;
            case "Right Click Me":
                actions.contextClick(demoQaPage.rightClickButton).perform();
```

```
                break;
            case "Click Me":
                actions.click(demoQaPage.regularClickButton).perform();
                break;
            default:
                throw new NoSuchElementException("No such button name as " + buttonName);

        }
        WaitHelper.shortWait();
        }
    @Then("user can see a message {string}")
    public void user_can_see_a_message(String expectedMessage) {
//        String actualMessage = demoQaPage.doubleClickMessage.getText();
//        Assert.assertEquals("Message verification failed",expectedMessage,actualMessage);

         //USING SWITCH
        String actualMessage;
        switch (buttonName) {
            case "Double Click Me":
                actualMessage = demoQaPage.doubleClickMessage.getText();
                Assert.assertEquals(  "Message verification failed", expectedMessage,
actualMessage);
                break;
            case "Right Click Me":
                actualMessage = demoQaPage.rightClickMessage.getText();
                Assert.assertEquals( "Message verification failed", expectedMessage,
actualMessage);
                break;
            case "Click Me":
                actualMessage = demoQaPage.regularClickMessage.getText();
                Assert.assertEquals(  "Message verification failed", expectedMessage,
actualMessage);
                break;
    }
          // USING IF STATEMENT
        if(buttonName.equals("Double Click Me")){
            actualMessage = demoQaPage.doubleClickMessage.getText();
            Assert.assertEquals(  "Message verification failed", expectedMessage,
actualMessage); }
        else if (buttonName.equals("Right Click Me")) {
            actualMessage = demoQaPage.rightClickMessage.getText();
            Assert.assertEquals(  "Message verification failed", expectedMessage,
actualMessage); }
        else if (buttonName.equals("Click Me")){
        }
        actualMessage = demoQaPage.regularClickMessage.getText();
        Assert.assertEquals(  "Message verification failed", expectedMessage,
actualMessage);
    }

}


package pages;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
```

```
import utilities.Driver;

public class DemoQaPage {
    public DemoQaPage(){
        PageFactory.initElements(Driver.getInstance().getDriver(),this);

    }
    @FindBy(id = "doubleClickBtn")
    public  WebElement doubleClickButton;
    @FindBy(id = "rightClickBtn")
    public WebElement rightClickButton;
    @FindBy(xpath = "//*[text() = 'Click Me']")
    public WebElement regularClickButton;

    @FindBy(id = "doubleClickMessage")
    public WebElement doubleClickMessage;
    @FindBy(id = "rightClickMessage")
    public WebElement rightClickMessage;
    @FindBy(id = "dynamicClickMessage")
    public WebElement regularClickMessage;
}

package utilities;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class Driver {

    /* private constructor, no objects can be created outside of this class */
    private Driver(){

    }

    /* private static reference variable to the class */
    private static Driver instance=new Driver();

    /* create a getter method with return type of the class => encapsulation */
    public static Driver getInstance(){
        return instance;
    }
    private static ThreadLocal<WebDriver> threadLocalDriver = new ThreadLocal<>();

    public static WebDriver getDriver() {
        // Initialize the driver if it is null
        if (threadLocalDriver.get() == null) {
            System.setProperty("webdriver.chrome.driver",
"src/test/resources/drivers/chromedriver.exe"); // Set the path to your chromedriver
            ChromeOptions options = new ChromeOptions();
            options.addArguments("--start-maximized");
            threadLocalDriver.set(new ChromeDriver(options));
        }
        return threadLocalDriver.get();
    }

    public static void closeDriver() {
```

```
        if (threadLocalDriver.get() != null) {
            threadLocalDriver.get().quit();
            threadLocalDriver.remove();
        }
    }



}
```

```
package runner;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {"pretty", "html:target/reports/cucumber-
reports.html","json:target/cucumber.json"},
        features = "src/test/resources/features",
        glue = "step_definitions",
        tags = "@actionButtons",
        dryRun = false,
        monochrome = true
)
public class TestRunner {
    // No code is needed here; Cucumber will handle the execution
}
```

```
package utilities;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class ConfigurationReader {
    private static Properties properties;

    public static String getProperty(String fileName, String key) {
        if (properties == null) {
            properties = new Properties();
        }

        try {
            // Add file separator for proper path construction
            String path = System.getProperty("user.dir") + "/src/test/resources/features/"
+ fileName;

            // Debugging: Print the constructed path
            System.out.println("Loading properties from: " + path);

            // FileInputStream reads the properties file
            FileInputStream stream = new FileInputStream(path);
            properties.load(stream);
```

```
            stream.close();
        } catch (IOException e) {
            e.printStackTrace();
            throw new RuntimeException("Failed to load properties file: " + fileName);
        }

        return properties.getProperty(key);
    }
}
```

qa_configs.properties

```
mysql.url: jdbc:mysql://dev-hrm.yoll.io:3306/orangehrm_mysql
yollhrm.database.username: root
yollhrm.database.password: Bootcamp5#
mb.database.url: jdbc:sqlserver://database-1.cd2kmrv4yrcl.us-east-
1.rds.amazonaws.com;databaseName=mealb
mb.database.username: student
mb.database.password: Student2022Q
mealb.url: https://dev-mb.yoll.io/Account/Login
mealb.username: walmart
mealb.password: Mart12345

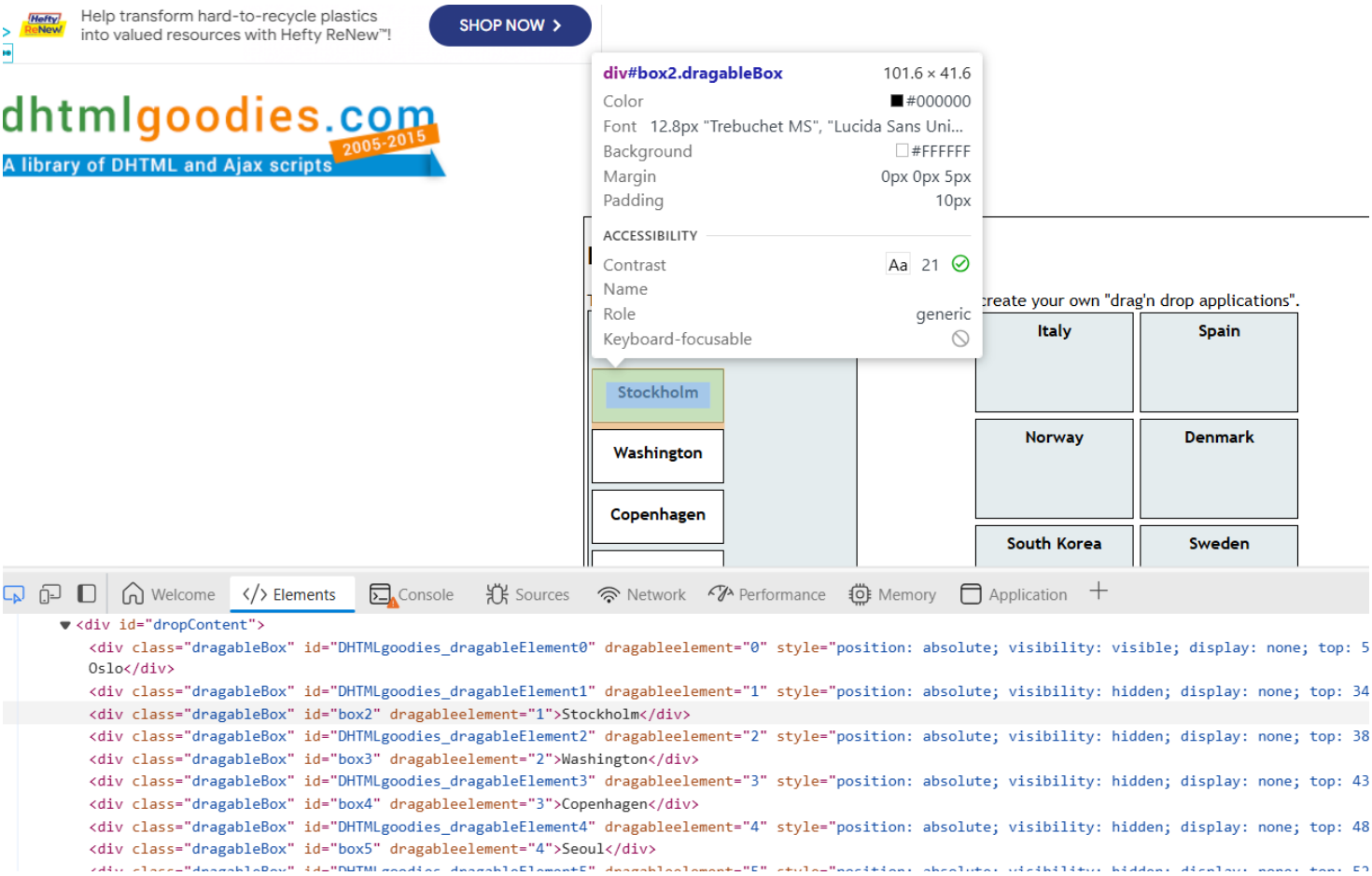
#HRM URL and valid credentials
yollhrm.url: http://dev-hrm.yoll.io/index.php/auth/login
yollhrm.username:yoll-student
yollhrm.password:Bootcamp5#

#Alert Page URL
alert.url: https://demoqa.com/alerts

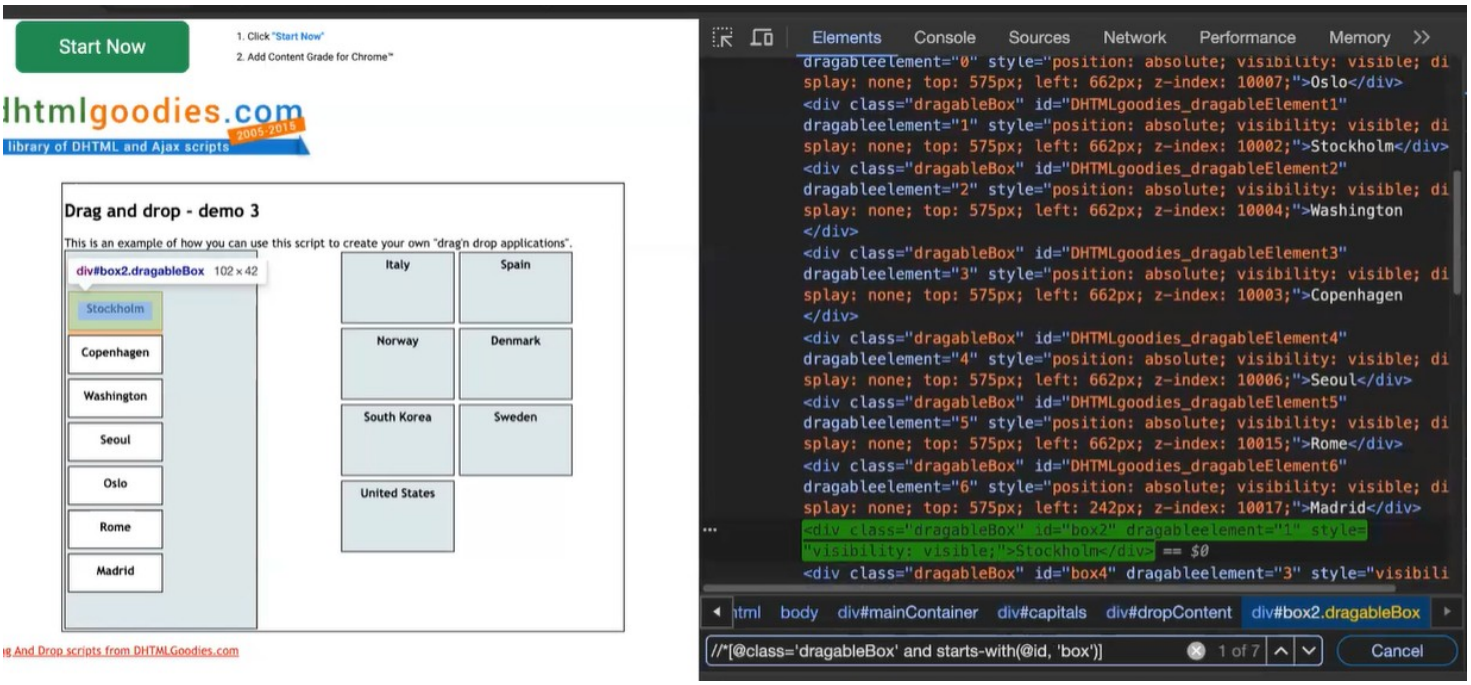
#Buttons Page URL
demoqa.url: https://demoqa.com/buttons

#iFrames Page URL
iFrames.url: https://the-internet.herokuapp.com/iframe
```

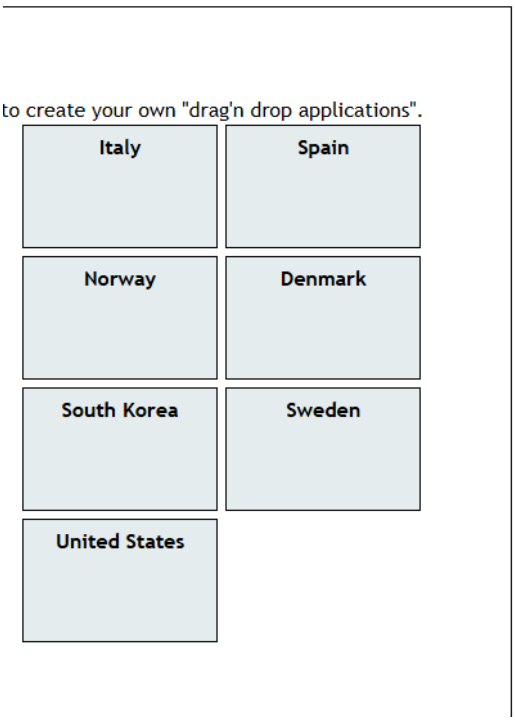
Capitals:



By using the research bar, **we select the dragableBox class that starts with an id containing box: 7 results**

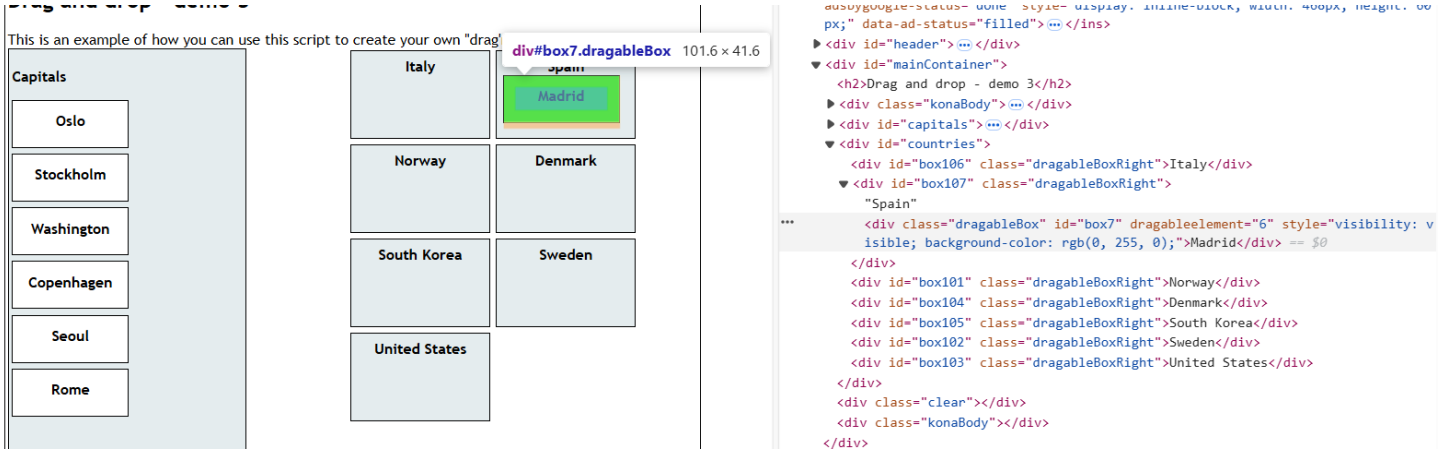


Countries:




```
show_ads.js"> </script>
<ins class="adsbygoogle adsbygoogle-noablate" data-ad-client="pub-07142364
63" data-ad-height="60" data-ad-slot="7945138075" data-ad-width="468" data
adsbygoogle-status="done" style="display: inline-block; width: 468px; heig
px;" data-ad-status="filled">...</ins>
<div id="header">...</div>
<div id="mainContainer">
  <h2>Drag and drop - demo 3</h2>
  <div class="konaBody">...</div>
  <div id="capitals">...</div>
  <div id="countries">
    <div id="box106" class="dragableBoxRight">Italy</div> == $0
    <div id="box107" class="dragableBoxRight">Spain</div>
    <div id="box101" class="dragableBoxRight">Norway</div>
    <div id="box104" class="dragableBoxRight">Denmark</div>
    <div id="box105" class="dragableBoxRight">South Korea</div>
    <div id="box102" class="dragableBoxRight">Sweden</div>
    <div id="box103" class="dragableBoxRight">United States</div>
  </div>
  <div class="clear"></div>
  <div class="konaBody"></div>
</div>
<div id="debug"></div>
<script type="text/javascript">...</script>
<p>...</p>
<ins class="adsbygoogle adsbygoogle-noablate" data-adsbygoogle-status="don
style="display: none !important;" data-ad-status="unfilled">...</ins>
```

After Draging Madrid to Spain:



First Method with only one Capital and Country:

```
Feature: Capitals and Countries Match
  @capitalsAndCountries
  Scenario: Drag and Drop capitals to its Countries
    Given user navigates to dhtmlgoodies countries page
    When user moves capital "Rome" to country "Italy"
    Then user can see capital box is green
```

Second Method with the whole Capitals and Countries:

```
Feature: Capitals and Countries Match

  @capitalsAndCountries
  Scenario Outline: Drag and Drop capitals to its Countries
    Given user navigates to dhtmlgoodies countries page
    When user moves capital "<Capital>" to country "<Country>"
    Then user can see capital box is green

    Examples:
      | Capital    | Country       |
      | Rome       | Italy         |
      | Norway     | Oslo          |
      | Stockholm  | Sweden        |
      | Washington | United States |
      | Copenhagen | Denmark       |
      | Seoul      | South Korea   |
      | Madrid     | Spain         |
```

#DragAndDrop URL
dragAndDrop.url: http://www.dhtmlgoodies.com/scripts/drag-drop-custom/demo-drag-drop-3.html

```
package step_definitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
```

```
import pages.CountriesPage;
import utilities.ConfigurationReader;
import utilities.Driver;
import utilities.WaitHelper;

import javax.swing.plaf.basic.BasicSliderUI;

public class CountriesPageSteps {
    private CountriesPage countriesPage =new CountriesPage();
    private Actions actions= new Actions(Driver.getInstance().getDriver());
    private String capital;
    @Given("user navigates to dhtmlgoodies countries page")
    public void user_navigates_to_dhtmlgoodies_countries_page() {
        String url = ConfigurationReader.getProperty("ui-
config.properties","dragAndDrop.url");
        Driver.getInstance().getDriver();
    }
    @When("user moves capital {string} to country {string}")
    public void user_moves_capital_to_country(String capital, String country) {
        this.capital=capital;
        WebElement sourceElement=countriesPage.getCapitalWebElementByName(capital);
        WebElement targetElement=countriesPage.getCountryWebElementByName(country);
        actions.dragAndDrop(sourceElement,targetElement).perform();
        WaitHelper.shortWait();

    }
    @Then("user can see capital box is green")
    public void user_can_see_capital_box_is_green() {
        String actualCapitalBoxColor
=countriesPage.getCapitalWebElementByName(capital).getCssValue("background-color");
        System.out.println("CURRENT COLOR: "+actualCapitalBoxColor);
        String expectedColor ="rgba(0,255,0,1)";
        Assert.assertEquals("Color verification
failed",expectedColor,actualCapitalBoxColor);
    }

}


package pages;

import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import utilities.Driver;

import java.util.List;

public class CountriesPage {
    public CountriesPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);

    }

    @FindBy(xpath = "//*[@class='dragableBox' and starts-with (@id, 'box')]")
    public List<WebElement> capitals;
```

```
    @FindBy(xpath = "//*[@class='dragableBoxRight']")
    public List<WebElement> countries;

    public WebElement getCapitalWebElementByName(String capitalName) {
        for (WebElement capital : capitals) {

            if (capital.getText().equals(capitalName)) {
                return capital;
            }
        }
            throw new NoSuchElementException("There is no such capital with name" +
capitalName);

        }

    public WebElement getCountryWebElementByName(String countryName) {
        for (WebElement country : countries) {

            if (country.getText().equals(countryName)) {
                return country;
            }
        }
        throw new NoSuchElementException("There is no such capital with name" +
countryName);

    }

    }


package runner;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {"pretty", "html:target/reports/cucumber-
reports.html","json:target/cucumber.json"},
        features = "src/test/resources/features",
        glue = "step_definitions",
        tags = "@capitalsAndCountries",
        dryRun = true,
        monochrome = true
)
public class TestRunner {
    // No code is needed here; Cucumber will handle the execution
}
```

BUILD()

- If we want to build series of actions methods, we use build() method
  - ```
    act
        .moveToElement(txtUsername)
        .click()
        .keyDown(txtUsername, Keys.SHIFT)
        .sendKeys(txtUsername, "hello")
        .keyUp(txtUsername, Keys.SHIFT)
        .build().perform() ;
    ```

```
Feature: Login with Multiple Actions
  @actionsLogin
  Scenario: Login with valid credentials with actions class
    Given user navigates to hrm login page
    When user logs in with username "yoll-student" and password "Bootcamp5#" with actions
class
    Then user is redirected to home page
```

```
package runner;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {"pretty", "html:target/reports/cucumber-
reports.html","json:target/cucumber.json"},
        features = "src/test/resources/features",
        glue = "step_definitions",
        tags = "@actionsLogin",
        dryRun = false,
        monochrome = true
)
public class TestRunner {
    // No code is needed here; Cucumber will handle the execution
}
```

```
package step_definitions;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
```
```

```
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.interactions.Actions;
import pages.LoginPage;
import utilities.Driver;
import utilities.WaitHelper;

public class OhrmLoginPageSteps {

    private WebDriver driver = Driver.getInstance().getDriver();
    private  LoginPage ohrmLoginPage= new LoginPage(driver);
    private Actions actions = new Actions(Driver.getInstance().getDriver());

    @Given("user navigates to hrm login page")
    public void user_navigates_to_hrm_login_page() {
        driver.get("http://dev-hrm.yoll.io/index.php/auth/login");
    }
    @When("user clicks login button")
    public void user_clicks_login_button() {
    ohrmLoginPage.loginButton.click();

    }

    @When ("user logs in with username {string} and password {string}")
    public void user_enters_username_and_password (String username, String password) {
    ohrmLoginPage.login(username, password);
    }

    @Then ("user can see error message {string}")
    public void user_can_see_error_message(String expectedErrorMessage) {
            String actualErrorMessage =
ohrmLoginPage.invalidCredentialsErrorMessage.getText();
            Assert.assertEquals ( "Error message verification failed",
expectedErrorMessage, actualErrorMessage);
}

    @When ("user logs in with username {string} and password {string} with actions class")
    public void user_logs_in_with_username_and_password_with_actions_class (String
username, String password) {
         actions.moveToElement (ohrmLoginPage.usernameInputBox).
                 click().
                 sendKeys (username).
                 sendKeys (Keys. TAB).
                 sendKeys (password).
                 moveToElement (ohrmLoginPage. loginButton).
                 click().
                 build().perform();
         WaitHelper.shortWait();
    }
}

package pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
```

```java
public class LoginPage {

    public LoginPage(WebDriver driver) {
        PageFactory.initElements(driver, this);
    }

    @FindBy(id = "txtUsername")
    public WebElement usernameInputBox;

    @FindBy(id = "txtPassword")
    public WebElement passwordInputBox;

    @FindBy(id = "btnLogin")
    public WebElement loginButton;

    @FindBy(id = "spanMessage")
    public WebElement invalidCredentialsErrorMessage;

    public void login(String username, String password){

        usernameInputBox.sendKeys(username);
        passwordInputBox.sendKeys(password);
        loginButton.click();
    }

}
```

Actions Class Keyboard Actions

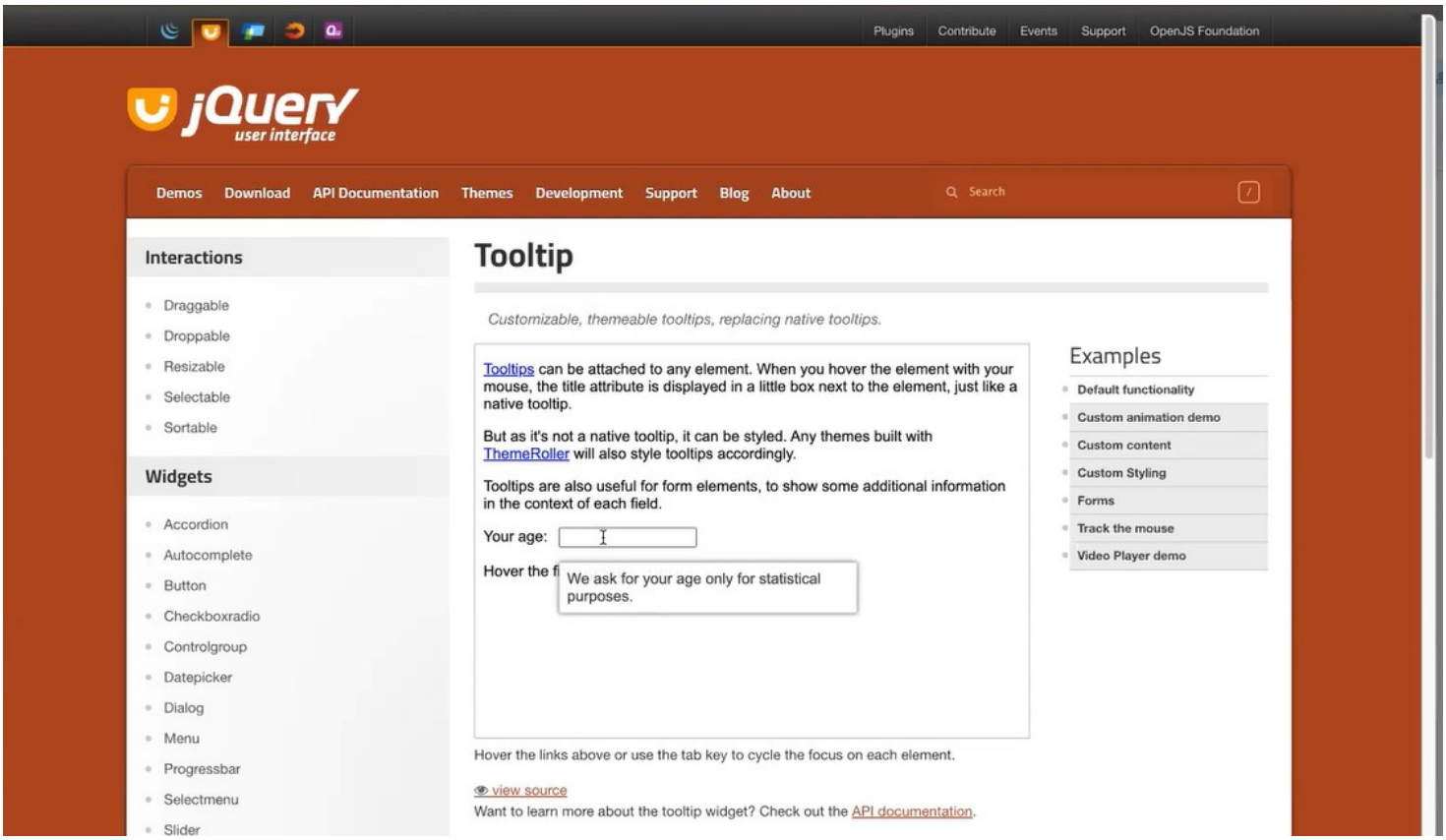
- Normally we need to find an inputtext to use sendKeys On
- We can also use Actions class to press keys on keyboard
  - `Actions act=new Actions(driver);`
  - `act.sendKeys(Keys.ENTER).build().perform();`

```
This is similar way to driver.findElement(By.locator).sendKeys(Keys.Enter);
```

- This will just press the button on the keyboard without using any element on the page.

Tooltip

- Tooltips show up when you hover over the element
- But to confirm tooltip is present, we can just use **title** attribute of the html element
  - `WebElement elem = driver.findElement(By.id("num"));`
  - `String tooltiptext=elem.getAttribute("title");`



Summary Knowledge Transfer

```
--------------------------------------------------------------------------------
THIS FILE WILL CONTAIN KNOWLEDGE TRANSFER(SHARING WITH INFORMATION)
KT - KNOWLEDGE TRANSFER is a common meeting for people who are new to the project and need to get started
--------------------------------------------------------------------------------
A. Explain in which case we use scenario outline and in which case we use the table?
```

```
    1A. Scenario Outline should have an Example table. This example table will have multiple sets of data.
    - We use Scenario Outline where we want to run a scenario from the perspective of multiple users that will use different sets of data.
    - The data that is in Example Table can be used across all Scenario.
    - The intention is that you run the same Scenario with multiple sets of data
    - A hint to recognize if a case is good for Scenario Outline would: if you can split your scenario outline in individual scenarios it is a good case

    2A. Data Table - when we have to provide data to our step in a dynamic way we could use parameters.
    - Parameters can be used to pass data from steps to java methods.
    - These parameters are usually a short data that can be a string number etc...
    - The question is what do we do when we need to pass some data to one step that is much more complex.
    - We use data table when our parameter becomes too complex to pass as a string or number
    - For example if you pass some registration info about a user
    -- If passing as a string we will have to apply some sort of separation
    -- Also, each time we will have to read and understand the data to be able to interact with it
    -- That's why we form a data table that will be our more complex parameter for current step
    -- Then we can structure the data in more readable way as a List, Map, List of Maps

B. Question is regarding repetitive steps in scenarios.
- Let's say we have 3 user stories and a couple of the steps in those stories are similar.
- What is the best way of executing those stories:
  -- a. extend the classes where we have the steps
  -- b. dump them to the steps that we already have, like accumulating them in one go as we did with ohrm home page or user login steps?

1B. The whole idea of cucumber when it comes to steps that they are reusable.
   - That's why we have parametrization
   - Data Tables
   - Scenario Outline
   - The idea is to make our steps as reusable as possible
   - When you write a scenario in an existing framework one of the golden rules will be stay DRY! - Donot Repeat Yourself!
```

```
C. ENUMS vs Config files
- ENUM is nothing than a Constant wrapped in a specific type
- Config is an external file that contains configs/settings/hosts/endpoints

- If we talk about base URL - it is more common to place in config files
- Endpoints are a candidate to be stored in any form - as constant or configuration
http://dev-hrm.yoll.io/
/dashboard
String DASHBOARD_ENDPOINT = "/dashboard"
```

Ro

User Story: Cucumber AC 2

Scenario: Missing Required Fields When Modifying Employee's Information

- **Given** user is logged in and on the "PIM" tab
- **When** user selects an existing employee
- **Then** user is on the Employee Information Page
- **When** the user clicks the Edit button and clears the First Name and Last Name fields
- **Then** a "Required" error message should be displayed under both the First Name and Last Name fields

- ***Note: Please extract more than 2 scenarios for this AC***
  1. Create a new employee and edit personal information of that employee
  2. Pick an existing employee from the table and edit one of the existing ones